Software Reuse in Safety-Critical Systems

Barbara Lingberg
Leanna Rierson
May 22, 2003

Acronyms

- AC — Advisory Circular
- CAST — Certification Authorities Software Team
- CNS — Communication-Navigation-Surveillance
- COTS — Commercial-off-the-shelf
- CSTA — Chief Scientific and Technical Advisor
- FAA — Federal Aviation Administration
- IEEE — Institute of Electrical and Electronics Engineers
- IMA — Integrated Modular Avionics
- OOT — Object-oriented Technology
- OS — Operating System
- RTOS — Real-Time Operating System
- TSO — Technical Standard Order

Overview

- Software Reuse:
  What It Is, Is Not, and Its Goal
- Benefits/Risks/Myths of Software Reuse
- Software Assurance in Civil Aviation
- Approaches to Software Reuse
- FAA Activities Related to Software Reuse
- Summary

Software Reuse:
What It Is

- Process of creating software systems from existing software assets, rather than building software systems from scratch (Krueger)

Software Reuse: *What It Is (cont)*

- Assets can be software components, objects, software requirement analysis and design models, domain architecture, database schema, code documentation, manuals, standards, test scenarios, and plans (Sodhi)
- May occur within a software system, across similar systems, or in widely different systems (Sodhi)

Software Reuse: *What It Is Not*

- Software Reuse != Software Salvaging (Adolf)
  - Software reuse is software that is designed to be reused
  - Software salvaging is using software that was not designed for reuse

Software Reuse:
Its Goal

- From previous development efforts
  - Use as much software data as possible
  - To reduce time, cost, and risks associated with re-development

Benefits of Software Reuse

- Meeting business needs
- Higher productivity
- Increased quality
- Quicker time to market
- Better use of resources
- Helps with system complexity issues

Risks of Software Reuse

- Requires upfront investment
- Is a gamble on the future
- Can end up costing more
- Can induce errors
- **Must be used cautiously in safety-critical domains**

Myths of Software Reuse

- Reuse is quick, easy, simple, & free
- Buying components means no building
- Components = reuse
- Reuse is only for code
- Maintenance != Development so reuse does not apply in maintenance
- Increased productivity means loss of jobs

Software Assurance in Civil Aviation

- RTCA/DO-178B "Software Considerations in Airborne Systems and Equipment Certification" is "defacto" guidance document
- Focuses on software aspects of system development
- Identifies software levels and objectives based on software contribution of failure conditions
- Used in aviation, CNS systems, military systems, medical equipment
- Reuse approaches evaluated against RTCA/DO-178B objectives

Approaches to Software Reuse

- Planning for Reuse
- Domain Engineering
- Software Components
- Object-Oriented Technology
- Portability
- Commercial-off-the-shelf (COTS) Software
- Product Service History

Approaches to Software Reuse: *Planning for Reuse*

- Reuse doesn't just happen – requires planning, management, and execution
- Planning should address:
  - Process for Reuse
  - Safety
  - Integration - Software/Software and Software/Hardware
  - Portability
  - Maintenance
  - Re-Verification

Approaches to Software Reuse: *Planning for Reuse (cont)*

- Keys to Success (McConnell)
  - Take advantage of personnel continuity between old and new programs
  - Do not overestimate savings
  - Secure long-term, high-level management commitment to a reuse program
  - Make reuse an integral part of the development process
  - Establish a separate reuse group
  - Focus on small, sharp, domain-specific components
  - Focus design efforts on abstraction & modularity

Approaches to Software Reuse: *Domain Engineering*

- Definition: Process of creating assets that can be managed and reused through
  - Domain analysis
  - Domain design
  - Domain implementation
- Domain is a group or family of related systems. All systems in that domain share a set of capabilities and/or data. (Sodhi)
- Domain engineering is a relatively immature field

Approaches to Software Reuse: *Domain Engineering (cont)*

- Offers greatest potential for productivity and quality gains through:
  - Knowledge reuse
  - Reuse of architectural domain knowledge
  - Repositories of components e.g., general-purpose libraries of software architectures
  - Reuse of software designs and patterns
  - Reduction of "cognitive distance"

Approaches to Software Reuse: *Software Components*

- What is a Software Component?
  - Prewritten elements of software with clear functionality and well-defined interface (Rhodes)
  - Software code and supporting RTCA/DO-178B documentation being considered for reuse. Forms a portion of the software that will be implemented by the integrator/applicant. (FAA Draft Advisory Circular)

Approaches to Software Reuse: *Software Components (cont)*

- Qualities (Meyer)
  - Careful specification of functionality & interface
  - Correctness - works as specified
  - Robustness - doesn't fail if used properly
  - Ease of identification
  - Ease of learning
  - Wide-spectrum of coverage
  - Consistency
  - Generality - useful for multiple environments
- Examples
  - Real-time Operating System (RTOS)
  - Software Libraries

Approaches to Software Reuse: *Software Components (cont)*

- Safety Concerns
  - Planning
  - Requirements Traceability
  - Re-verification
  - Interface documents
  - Partitioning/protection
  - Artifacts
  - Maintenance
  - Unused code

Approaches to Software Reuse: *Object-Oriented Technology*

- Definition: A software development technique in which a system or component is expressed in terms of objects and connections between those objects (IEEE)
- Centered around "classes" and "objects"
  - Class: set of objects that share a common structure and a common behavior (Booch)
  - Object: instance of a class

Approaches to Software Reuse: *Object-Oriented Technology (cont)*

- Benefits for Reuse
  - Breaks complex systems into manageable pieces
  - Easier to implement OO design into code
  - Supports use of development tools
- Safety Concerns
  - Dead/Deactivated Code
  - Dynamic Binding/Dispatch
  - Encapsulation
  - Inheritance
  - Polymorphism

Approaches to Software Reuse: *Portability*

- Goal: Transport software to new platforms and/or environments with minimal adaptation

Approaches to Software Reuse: *Portability (cont)*

- Strategy:
  - Identify minimum necessary set of environmental requirements & assumptions
  - Eliminate all unnecessary assumptions throughout the design
  - Identify specific environment interface required
  - Anticipate need to "bridge the gap" for environments which don't meet interface assumptions

Approaches to Software Reuse: *Portability (cont)*

- Concerns include:
  - Operating System inconsistencies
  - Different compiler options/effects
  - Incompatible libraries
  - Run-time problems
  - Underestimation of integration effort
  - Architectural inconsistency

Approaches to Software Reuse: *Commercial off the Shelf (COTS)*

- Definition:
  - Commercially available applications sold by vendors through public catalog listings.
  - COTS software is not intended to be customized or enhanced.
  - Contract-negotiated software developed for a specific application is not COTS software (RTCA/DO-178B)
- Common uses:
  - Operating systems (OS)
  - Real-time operating systems (RTOS)

Approaches to Software Reuse: *COTS (cont)*

- Concerns of COTS OS include:
  - Integrity of design and implementation may be unknown
  - Unknown functionality and side effects may exist
  - Negative effect on operation of other software applications executing using OS functions
  - Mitigation approaches may themselves be implemented in COTS operating system's environment
  - Unknown errors may exist
  - Difficulty in satisfying DO-178B objectives
  - Patches may have safety impact

Approaches to Software Reuse: *Product Service History*

- Definition: Contiguous period of time during which the software is operated within a known environment, and during which successive failures are recorded (RTCA/DO-178B)
- Purpose is to gain confidence in software over a period of time

Approaches to Software Reuse: *Product Service History (cont)*

- Considerations:
  - Configuration management of the software
  - Effectiveness of problem reporting
  - Stability and maturity of the software
  - Relevance of product service history environment
  - Actual error rates and product service history
  - Impact of modifications

Approaches to Software Reuse: *Product Service History (cont)*

- Attributes
  - Service duration length
  - Change control during service
  - Proposed use versus service use
  - Proposed environment versus service environment
  - Number of significant modifications during service
    - Hardware and software
  - Error detection and reporting capabilities
  - Number of in-service errors
  - Amount and quality of service history data available and reviewed

FAA Activities Related to Software Reuse

- "Software Approval Guidelines" Order, Chap. 12: *Reuse of software life cycle data*
- Reusable Software Component Advisory Circular (8110.RSC draft) – reuse of third party components
- TSO for Integrated Modular Avionics (IMA) Hardware Elements – TSO-C153

FAA Activities Related to Software Reuse (cont)

- IMA Advisory Circular – AC-145
- RTCA Special Committee #200
- Service History Handbook
- CAST Papers
- COTS Research Project
- OO Technology in Aviation Handbook

Summary

- Reuse requires planning
- Techniques and tools exist to help
- Tips for success:
  - Obtain top level management support
  - Overcome non-technical inhibitors
  - Make reuse integral to development process
  - Focus on domain-specific components
  - Develop reuse guidelines and measurements
- Safety must be a priority
- FAA has several initiatives underway to enable reuse

For More Information

Leanna Rierson

Chief Scientific and Technical Advisor for Aircraft Computer Software

FAA/AIR-106N

Leanna.Rierson@faa.gov

Barbara Lingberg

Software Program Manager

FAA/AIR-120

Barbara.Lingberg@faa.gov

Software Website: http://av-info.faa.gov/software